


REMARKS

The above amendments and following remarks are submitted in response to the Official Action of the Examiner mailed December 18, 2002. Having addressed all objections and grounds of rejection, claims 1-20, being all the pending claims, are now deemed in condition for allowance. Reconsideration to that end is respectfully requested.

The Examiner has objected to the specification and Abstract as containing a number of informalities. In response thereto, Applicants have amended the specification and abstract as required. These above amendments as supported by Appendix A are deemed to fully address these objections.



The present invention as disclosed and claimed involves a "flush buffer" which accompanies a "store-in" cache memory. A store-in cache memory is one in which operands stored from a processor are not immediately and automatically stored to lower level memory. This is in direct contrast to "store-through" cache memories from which operands stored from a processor are immediately and automatically stored in lower level memory.

Those of skill in the art are able to suggest the advantages of one cache configuration over another based upon various system variables. For example, "store-in" cache memories tend to be more efficient for relatively large size cache memories when repetitive operations are performed upon a relatively small amount of data,

because a single datum may be modified many times without the need to store into lower level memory. Similarly, a store-through cache memory might be more efficient for a small sized cache memory whenever any given datum is only modified very infrequently.

The flushing process of a store-in cache memory stores data at a lower level memory whenever modified data must be removed from the store-in cache memory to make room for data newly requested by a processor. Without the flushing process, that modified data would be lost to the system, because it would not otherwise find its way into the lower level memory.

Store-through cache memories do not require any flushing process, because any modified data is immediately and automatically written to lower level memory. This occurs as a matter of design by definition.

In making his rejections, the Examiner has apparently confused these basic concepts. For his convenience, Applicants have provided a discussion of these matters at page 7, line 6, through page 8, line 4, of the specification.

The Examiner has rejected claims 1, 6, 7, 11, and 16 under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,276,848, issued to Gallagher et al (hereinafter referred to as "Gallagher"). This ground of rejection is respectfully traversed for the reasons provided below.

"It is axiomatic that for prior art to anticipate under §102 it has to meet every element of the claimed invention, and that

such a determination is one of fact." *Hybritech Inc. v. Monoclonal Antibodies, Inc.*, 802 F.2d 1367, 231 USPQ 81, 90 (Fed. Cir. 1986).

The rejection of claims 1, 6, 7, 11, and 16 is respectfully traversed, because Gallagher does not "meet every element of the claimed invention". In making his rejection, the Examiner states:

As per claims 1, 6 and 16, Gallagher discloses a data processing system having a processor responsively coupled to a store-in cache memory which is responsively coupled to a lower level memory (i.e., processor 20, 22; cache 26a, memory 10b) [Fig. 2; col. 3, lines 25-58],

Thus, the Examiner has read a portion of claim 1 on to the second embodiment of Gallagher which is identified at column 3, line 24-25, which states:

Referring to FIG. 2, a triadic (multiprocessor) system is illustrated.

It is important to note, that even though this embodiment has a store-in cache memory, it does not have a flush buffer.

The Examiner continues his rejection by stating:

....the improvement comprising a flush buffer responsively coupled to said store-in cache memory and said lower level memory (i.e., store buffer in SCL 12 positioned between L1 and L3) [Fig. 1; col. 2, line 66-col. 3, line 5].

The Examiner has read the second portion of the claim on to the first embodiment of Gallagher. It is clear that this is a different embodiment from column 2, lines 11-12, which states:

Referring to FIG. 1, a uniprocessor computer system of the present invention is illustrated.

In addition to the impermissible combining of mutually exclusive different embodiments, it is also important to note that this first

embodiment has a "store-through" cache memory as its only data cache memory. Column 2, lines 66-67, states:

The data cache 18 is a "store through" cache.....
Because the cache memory of this first embodiment is a "store-through" cache memory, it has no need for a flush process. That also means that it has no need for a flush buffer. In an attempt to make his rejection, the Examiner clearly erroneously says that the "store buffer" is located in SCL 12. Column 3, lines 4-5, states:

.... a "store buffer" is present with the L1 data cache 18....

Furthermore, the function of the "store buffer" is not clear from the text (it is not shown in the drawings). Column 2, line 66, through column 3, line 6, curiously states:

The data cache 18 is a "store through" cache, which means that data from the instruction/execution units 20 are stored in L3 memory and, if the corresponding obsolete data is not present in the L1 caches 18, the data is not brought into and stored in the L1 caches. To assist this operation, a "store buffer" is present with the L1 data cache 18 which is capable of buffering up to 8 store operations.

It is not clear what operation is assisted by the "store buffer". However, it cannot be a flush process, because data cache 18 is a store through cache memory.

Therefore, the rejection of claims 1, 6, and 16 is respectfully traversed. Furthermore, the rejection of claim 16 is deemed inadequate as a matter of law for failure to comply with MPEP 2181, et seq. Claim 16 contains "means-plus-function"

limitations which have not been addressed in accordance with mandated examination procedures.

Claim 7 depends from claim 6 and is further limited by a logical division within the flush buffer. Gallagher does not teach or suggest a flush buffer. In addition, there is no showing that the alleged "store buffer" of the first embodiment of Gallagher is divided in any fashion.

Claim 11 is an independent method claim having "experiencing", "selecting", and "transferring" steps which are not found in Gallagher. Instead, the Examiner inexplicably recites the incomprehensible language of column 3, lines 1-6. The rejection of claim 11 is respectfully traversed.

Thus, the rejection of claims 1, 6-7, 11, and 16, and all claims depending therefrom, is respectfully traversed.

The Examiner has rejected claims 2-5, 8-10, 12-15, and 17-20 under 35 U.S.C. 103(a) as being unpatentable over Gallagher in view of U.S. Patent No. 6,460,114, issued to Jeddeloh (hereinafter referred to as "Jeddeloh"). This rejection is respectfully traversed for failure of the Examiner to present a *prima facie* case of obviousness as required by MPEP 2143. This provision requires the Examiner to show: 1) motivation for alleged combination; 2) reasonable likelihood of success of alleged combination; and 3) all claimed elements present in the alleged combination.

The Examiner has not even bothered to address the requirement to show reasonable likelihood of success. He has only made a cursory attempt at showing motivation, and has instead appeared to

concentrate his efforts on finding the pieces of Applicants' invention wherever he can.

The Examiner states:

As per claim 2 and 12, Gallagher discloses the claimed invention as detailed above in the previous paragraphs. However, Gallagher does not specifically teach a tag memory indicating whether a particular memory location has been modified as recited in the claim.

Of course not. One does not need a tag for store-through cache memories, such as data cache 18. It is only needed for store-in cache memories. Such tags would be superfluous with store-through cache memories. Similarly, Gallagher does not have a flush buffer. A flush buffer is superfluous with a store-through memory.

Nevertheless, the Examiner makes the nonsensical statement:

It would have been obvious to one of ordinary skill in the art, having the teachings of Gallagher and Jeddeloh before him at the time the invention was made, to modify the system of Gallagher to include a tag memory indicating whether a particular memory location has been modified and loading said flush buffer with data from said particular location within said store-in cache memory in response to said indication that said particular location has been modified because it would have provided better memory access transactions control as taught by Jeddeloh by reducing the read latency associated with the search for target data [col. 1, lines 59-60, col. 2, line 40] as taught by Jeddeloh.

The argument of the Examiner does not make sense and is not consistent with the disclosure of Gallagher. In addition, this is precisely the unsupported conclusion attacked by the Court of Appeals for the Federal Circuit stating in part:

Broad conclusory statements regarding the teaching of multiple references, standing alone, are not "evidence". *In re Dembiczak*, 175 F.3d 994, 50 U.S.P.Q. 2d 1614 (Fed. Cir. 1999).

Therefore, the rejection is respectfully traversed for failure to show motivation as well as the other two required showings.

In rejecting claims 4 and 18, the Examiner states:

As per claims 4 and 18, Gallagher discloses a flush buffer comprises a first flush buffer store and a second flush buffer store (i.e., buffers up to 8 stores) [col. 3, lines 4-6].

Again, the Examiner equates the "store buffer" of Gallagher, whatever that is, to the claimed flush buffer, with the foreknowledge that the reference employs a store-through data cache 18 requiring no flushing and therefore not needing a flush buffer. He then somehow finds the reference to have divided the "store buffer" into two portions. The rejection of claims 4 and 18 is respectfully traversed as not logically consistent.

In rejecting claims 5 and 19, the Examiner does not apply the prior art to the claim limitations but instead finds: "Jeddeloh discloses the concept of a temporary register....". The rejection of claims 5 and 19 is respectfully traversed as being incomplete.

In rejecting claim 8, the Examiner admits that Gallagher does not teach a temporary register. However, he finds that Jeddeloh does stating:

Jeddeloh discloses the concept of a temporary register coupled to a store-in cache memory, a first flush buffer store and a second flush buffer store (i.e., posted write buffer used as temporary staging area) [co. 3, lines 35-64].

Yet, the Examiner does not even allege that Jeddeloh has "a store-in cache memory, a first flush buffer store and a second flush buffer store". Though he has alleged that Gallagher does, he hasn't shown how a list of parts in Gallagher and a list of parts

in Jeddeloh are somehow coupled as in the claimed invention. The rejection of claim 8 is respectfully traversed as not logically consistent.

In rejecting claim 10, the Examiner clearly erroneously equates the claimed "logic circuit....which routes data...", with routing of "dirty cache lines". The rejection of claim 10 is respectfully traversed as based upon a clearly erroneous finding of fact.

In attempting to motivate the alleged combination in the rejection of claims 13-15, the Examiner states:

It would have been obvious.....to modify the system of Gallagher to include discloses (sic) inhibiting said transferring step....because it could have reduced the memory latency time experienced by the CPU as taught by Jeddeloh by selecting an existing cache line for replacement based on a status indication [co. 2, lines 1-24] as taught by Jeddeloh.

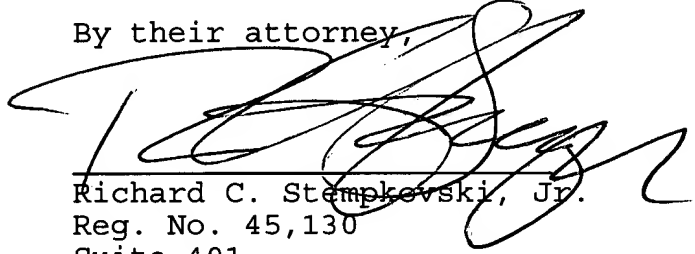
Yet, Gallagher does not disclose any cache lines. It is not clear whether the Examiner is suggesting that Gallagher should also employ such cache lines in store-through data cache 18. The rejection of claims 13-15 is respectfully traversed as being incomplete.

Having thus responded to each objection and ground of rejection, Applicants respectfully request entry of this amendment and allowance of claims 1-20, being the only pending claims.

Respectfully submitted,

Donald C. Englin et al

By their attorney,

A large, stylized handwritten signature in black ink, likely belonging to Richard C. Stempkowski, Jr., is written over the printed name and address.

Date March 18, 2003

Richard C. Stempkowski, Jr.
Reg. No. 45,130
Suite 401
Broadway Place East
3433 Broadway Street N.E.
Minneapolis, Minnesota
55413
(612) 331-1464

APPENDIX A (Support for Specification Amendments)

Please amend the specification and Abstract as follows:

1. At page 1, line 6, replace "_____", filed _____",
with - 09/651,598, filed August 30, 2000-;
2. At page 1, line 9, replace "_____", filed _____",
with - 09/650,800, filed August 30, 2000-;
3. At page 1, line 11, replace "_____", filed _____",
with - 09/651,597, filed August 30, 2000-;
4. At page 1, line 13, replace "_____", filed _____",
with - 09/650,730, filed August 30, 2000-;
5. At page 1, lines 14-15, replace "_____", filed _____
_", with - 08/235,196, filed April 29, 1994-;
6. At page 6, line 10, replace "pod"s", with -"pods";
7. At page 12, line 4, replace "TLC" with -Third Level Cache
(TLC)-;
8. At page 13, line 18, replace "UPI" with -Uninterruptible Power
Interface (UPI)-;
9. At page 13, line 20, replace "SLC ram", with -Second Level
Cache (SLC) RAM-;
10. At page 13, line 23, replace "FLC-IC" with -First Level Cache-
Instruction Cache-;
11. At page 14, line 1, replace "FLC-OC" with -First Level Cache-
Operand Cache-;

12. At page 14, line 3, replace "discusses", with -discussion-;
and

13. In the Abstract at line 11, replace "to be flush", with -to be
flushed-.

METHOD FOR MANAGING FLUSHES WITH THE CACHE

CROSS REFERENCE TO CO-PENDING APPLICATIONS

5 The present application is related to co-pending U.S. Patent
Application Serial No. 09/651,598, filed August 30, 2000, entitled
Cooperative Hardware and Microcode Control System for Pipelined
Instruction Execution; U.S. Patent Application Serial No.
09/650,800, filed August 30, 2000, entitled Method for Improved
10 First Level Cache Coherency; U.S. Patent Application Serial No.
09/651,597, filed August 30, 2000, entitled Method for Avoiding
Delays During SNOOP Requests; U.S. Patent Application Serial No.
09/650,730, filed August 30, 2000, entitled Leaky Cache Mechanism;
and U.S. Patent Application Serial No. 08/235,196, filed April 29,
15 1994, entitled Data Coherency Protocol for Multi-level Cached High
Performance Multiprocessor System, assigned to the assignee of the
present invention and incorporated herein by reference.

BACKGROUND OF THE INVENTION

20 1. Field of the Invention: - The present invention relates
generally to data processing systems employing multiple instruction
processors and more particularly relates to multiprocessor data
processing systems employing multiple levels of cache memory.

 2. Description of the Prior Art: - It is known in the art
25 that the use of multiple instruction processors operating out of

common memory can produce problems associated with the processing of obsolete memory data by a first processor after that memory data has been updated by a second processor. The first attempts at solving this problem tended to use logic to lock processors out of memory spaces being updated. Though this is appropriate for rudimentary applications, as systems become more complex, the additional hardware and/or operating time required for the setting and releasing of locks can not be justified, except for security purposes. Furthermore, reliance on such locks directly prohibits certain types of applications such as parallel processing.

The use of hierarchical memory systems tends to further compound the problem of data obsolescence. U.S. Patent No. 4,056,844 issued to Izumi shows a rather early approach to a solution. The system of Izumi utilizes a buffer memory dedicated to each of the processors in the system. Each processor accesses a buffer address array to determine if a particular data element is present in its buffer memory. An additional bit is added to the buffer address array to indicate invalidity of the corresponding data stored in the buffer memory. A set invalidity bit indicates that the main storage has been altered at that location since loading of the buffer memory. The validity bits are set in accordance with the memory store cycle of each processor.

U.S. Patent No. 4,349,871 issued to Lary describes a bussed architecture having multiple processing elements, each having a dedicated cache memory. According to the Lary design, each processing unit manages its own cache by monitoring the memory bus.

PMV
12/13/02

12/13/02
PMV

Any invalidation of locally stored data is tagged to prevent use of obsolete data. The overhead associated with this approach is partially mitigated by the use of special purpose hardware and through interleaving the validity determination with memory
5 accesses within the pipeline. Interleaving of invalidity determination is also employed in U.S. Patent No. 4,525,777 issued to Webster et al.

12/13/02
PMV

Similar bussed approaches are shown in U.S. Patent No. 4,843,542 issued to Dashiell et al, and in U.S. Patent No.
10 4,755,930 issued to Wilson, Jr. et al. In employing each of these techniques, the individual processor has primary responsibility for monitoring the memory bus to maintain currency of its own cache data. U.S. Patent No. 4,860,192 issued to Sachs et al, also employs a bussed architecture but partitions the local cache memory
15 into instruction and operand modules.

12/13/02
PMV

U.S. Patent No. 5,025,365 issued to Mathur et al, provides a much enhanced architecture for the basic bussed approach. In Mathur et al, as with the other bussed systems, each processing element has a dedicated cache resource. Similarly, the cache
20 resource is responsible for monitoring the system bus for any collateral memory accesses which would invalidate local data. Mathur et al, provide a special snooping protocol which improves system throughput by updating local directories at times not necessarily coincident with cache accesses. Coherency is assured
25 by the timing and protocol of the bus in conjunction with timing of the operation of the processing element.

12/13/02
PMV

An approach to the design of an integrated cache chip is shown in U.S. Patent No. 5,025,366 issued to Baror. This device provides the cache memory and the control circuitry in a single package. The technique lends itself primarily to bussed architectures. U.S. Patent No. 4,794,521 issued to Ziegler et al, shows a similar approach on a larger scale. The Ziegler et al, design permits an individual cache to interleave requests from multiple processors. This design resolves the data obsolescence issue by not dedicating cache memory to individual processors. Unfortunately, this provides a performance penalty in many applications because it tends to produce queuing of requests at a given cache module.

12/13/02
PMV

The use of a hierarchical memory system in a multiprocessor environment is also shown in U.S. Patent No. 4,442,487 issued to Fletcher et al. In this approach, each processor has dedicated and shared caches at both the L1 or level closest to the processor and at the L2 or intermediate level. Memory is managed by permitting more than one processor to operate upon a single data block only when that data block is placed in shared cache. Data blocks in dedicated or private cache are essentially locked out until placed within a shared memory element. System level memory management is accomplished by a storage control element through which all requests to shared main memory (i.e. L3 level) are routed. An apparent improvement to this approach is shown in U.S. Patent No. 4,807,110 issued to Pomerene et al. This improvement provides prefetching of data through the use of a shadow directory.

12/13/02
PMV

12/13/02
PMV

A further improvement to Fletcher et al, is seen in U.S. Patent No. 5,023,776 issued to Gregor. In this system, performance can be enhanced through the use of store around L1 caches used along with special write buffers at the L2 intermediate level.

12/13/02
p11v

5 This approach appears to require substantial additional hardware and entails yet more functions for the system storage controller.

Inherent in architectures which employ cache memory, is that the storage capacity is substantially less than the memory located at lower levels in the hierarchy. As a result, memory locations within the cache memory must often be cleared for use by other data quantities more recently needed by the instruction processor. For store-in cache memories, this means that those quantities modified by the instruction processor must first be rewritten to system memory before the corresponding location is available to store newly requested data. This "flushing" process tends to delay the availability of the newly requested data.

10

15

SUMMARY OF THE INVENTION

The present invention overcomes the problems found in the prior art by providing a method of and apparatus for improving the efficiency of cache memory within a system. This enhancement to efficiency is accomplished through a novel technique for managing the flushing process.

The preferred mode of the present invention includes up to four main memory storage units. Each is coupled directly to each of up to four "pods". Each pod contains a level three cache memory coupled to each of the main memory storage units. Each pod may also accommodate up to two input/output modules.

Each pod may contain up to two sub-pods, wherein each sub-pod may contain up to two instruction processors. Each instruction processor has two separate level one cache memories (one for instructions and one for operands) coupled through a dedicated system controller, having a second level cache memory, to the level three cache memory of the pod.

Unlike many prior art systems, both level one and level two cache memories are dedicated to an instruction processor within the preferred mode of the present invention. The level one cache memories are of two types. Each instruction processor has an instruction cache memory and an operand cache memory. The instruction cache memory is a read-only cache memory primarily having sequential access. The level one operand cache memory has read/write capability. In the read mode, it functions much as the

level one instruction cache memory. In the write mode, it is a semi-store-in cache memory, because the level two cache memory is also dedicated to the instruction processor.

In accordance with the present invention, the level two cache memory is of the store-in type. Therefore, the most current value of an operand which is modified by the corresponding instruction processor is first located within the level two cache memory. When the replacement algorithm for the level two cache memory determines that the location of that operand must be made available for newly requested data, that operand must be "flushed" into the lower level memory to avoid a loss of the most current value.

Waiting for flushing of the old data before requesting the new data induces unacceptable latency. Therefore, according to the present invention, a flush buffer is provided for temporary storage of the old data during the flushing process. Though this temporary storage appears at first to be a mere extension to the level two storage capacity, it greatly enhances efficiency because the flush process really does not need to utilize the level two cache memory.

The old data is moved from the level two cache memory to the flush buffer as soon as the replacement algorithm has determined which data to move, and the newly requested data is requested from the lower level memory. The flush process subsequently occurs from the flush buffer to the lower level of memory without further reference to the level two cache. Furthermore, locations within the level two cache memory are made available for the newly

requested data well before that data has been made available from the lower level memory.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects of the present invention and many of the attendant advantages of the present invention will be readily appreciated as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, in which like reference numerals designate like parts throughout the figures thereof and wherein:

FIG. 1 is an overall block diagram of a fully populated system in accordance with the present invention;

FIG. 2 is a schematic block diagram of one pod;

FIG. 3 is a schematic block diagram of one instruction processor along with its dedicated system controller;

FIG. 4 is a detailed diagram of the flush process of the present invention; and

FIG. 5 is a detailed diagram showing the flush buffers of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1 is an overall block diagram of fully populated data
5 processing system according to the preferred mode of the present
invention. This corresponds to the architecture of a commercial
system of Unisys Corporation termed "Voyager".

The main memory of the system consists of up to four memory
storage units, MSU 10, MSU 12, MSU 14, and MSU 16. Being fully
10 modular, each of these four memory storage units is "stand-alone"
and independent of one another. Each has a separate point-to-point
dedicated bi-directional interface with up to four "pods", POD 18,
POD 20, POD 22, **POD** 24. Again, each of the up to four pods is
separate and independent of one another.

15 The contents of POD 20 are shown by way of example. For the
fully populated system, POD 18, POD 22, and POD 24 are identical to
POD 20. The interface between POD 20 and each of the four memory
storage units (i.e., MSU 10, MSU 12, MSU 14, and MSU 16), is via a
third level cache memory designated cached interface, CI 26, in
20 this view. CI 26 couples with two input/output controllers, I/O
Module 44 and I/O Module 46, and two sub-pods, SUB 28 and SUB 30.
A more detailed explanation of the POD 20 is provided below.

The above described components are the major data handling
elements of the system. In the fully populated system shown, there
25 are sufficient components of each type, such that no single

hardware failure will render the complete system inoperative. The software employed within the preferred mode of the present system utilizes these multiple components to provide enhanced reliability for long term operation.

5 The remaining system components are utilitarian rather than data handling. System Oscillator 32 is the primary system time and clocking standard. Management System 34 controls system testing, maintenance, and configuration. Power Controller 36 provides the required electrical power. System Oscillator 38, Management System
10 40, and Power Controller 42 provide completely redundant backup capability.

FIG. 2 is a more detailed block diagram of POD 20. The level three cache memory interfaces directly with the memory storage units via Third Level Cache (TLC) Controller 26 (see also Fig. 1).
5 The actual storage for the level three cache memory is TLC SRAMS 48. As indicated this static random access memory consists of eight 16 byte memory chips.

Subpod 28 and subpod 30 each contain up to two individual instruction processors. These are designated Voyager IP 50,
10 Voyager IP 52, Voyager IP 54, and Voyager IP 56. As explained in detail below, each contains its own system controller. In accordance with the preferred mode of the present invention, these instruction processors need not all contain an identical software architecture.

FIG. 3 is a more detailed block diagram of Voyager IP 50, located within Subpod 28, located within POD 20 (see also Figs. 1 and 2). As explained above, each instruction processor has a dedicated system controller having a dedicated level two cache memory. Instruction processor 64 has two dedicated level one cache memories (not shown in this view). One level one cache memory is a read-only memory for program instruction storage. Instruction processor 64 executes its instructions from this level one cache memory. The other level one cache memory (also not shown in this view) is a read/write memory for operand storage.

Instruction processor 64 is coupled via its two level one cache memories and dedicated system controller 58 to the remainder of the system. System controller 58 contains input logic 74 to interface with instruction processor 64. In addition, data path logic 70 controls movement of the data through system controller 58. The utilitarian functions are provided by Locks, Dayclocks, and Uninterruptible Power Interface (UPI) 62.

The remaining elements of system controller 58 provide the level two cache memory functions. Second Level Cache (SLC) data RAM 66 is the data actual storage facility. Control logic 70 provides the cache management function. SLC tags 72 are the tags associated with the level two cache memory. First Level Cache-Instruction Cache (FLC-IC) Dup. Tags 76 provides the duplicate tags for the level one instruction cache memory of instruction processor 64. Similarly, First Level Cache-Operand Cache (FLC-OC) Dup. Tags

78 provides the duplicate tags for the level one operand cache memory of instruction processor 64. For a more complete discussion of this duplicate tag approach, reference may be made with the above identified co-pending and incorporated U.S. Patent Applications.

5

FIG. 4 is a detailed functional diagram showing the flushing process of the preferred mode of the present invention. Following a level one cache memory miss, a data request is made from level one operand cache memory 114 of instruction processor 110 (see also Fig. 3). In accordance with the present invention, the data request is made on memory bus 118.

If the requested data is found within second level cache memory 122 (i.e., a cache hit), the data access occurs. However, if a cache miss occurs within second level cache memory 122 (i.e., the data is not present), a level three cache memory request is made via path 178 and memory bus 130. As soon as the data is available, it is transferred from memory bus 130 via path 180.

To provide a place to store the newly requested data, cache memory 122 may need to flush some older data, if all locations are full. The selection of which location(s) to flush is in accordance with a least recently used algorithm as modified in accordance with the above identified and incorporated co-pending patent applications. The data to be flushed is transferred to flush buffer 186 from which the data is rewritten to level three memory via bus 130. Because this data is flushed from level two cache memory 122 to flush buffer 186 before the rewrite can be accomplished, space becomes quickly available within level two cache memory 122 for accommodating the newly requested data as soon as available.

FIG. 5 is detailed diagram showing the data flow in accordance with the present invention. Upon being notified of a level two cache miss, priority logic 188 determines which locations are to be flushed. This selection is made in the manner discussed above. The location(s) to be flushed is communicated to tag RAM 190 and data RAM 196 via addressing path 192.

Access of tag RAM 190 provides a determination whether there has been any modification to the data within level two cache memory. If there has been no modification as noted within tag RAM 190, no further write operation to level three memory is required. If the data has been modified, however, path 194 notifies priority logic 188 that the modified data to be flushed must be rewritten to level three memory.

Assuming that a rewrite is necessary, the data is accessed from data RAM 196 and transferred via path 200 to temp register 198. Further latency is reduced by employing two flush buffers (i.e., flush buffer0 132 and flush buffer1 134) as shown. Temp register 198 routes the data to be rewritten to either flush buffer0 132 via path 202 or to flush buffer1 134 as each becomes available.

The data to be flushed is stored within the selected flush buffer while the rewriting process is accomplished. The data to be transferred to level three memory via path 136 and bus 130.

Having thus described the preferred embodiments in sufficient detail for those of skill in the art to make and use the present invention, those of skill in the art will be readily able to apply the teachings found herein to yet other embodiments within the scope of the claims hereto attached.

WE CLAIM:

METHOD FOR MANAGING FLUSHES WITH THE CACHE

ABSTRACT OF THE DISCLOSURE

5 A method of and apparatus for improving the efficiency of a
data processing system employing a multiple level cache memory
system. The efficiencies result from managing the process of
flushing old data from the second level cache memory. In the
present invention, the second level cache memory is a store-in
10 memory. Therefore, when data is to be deleted from the second
level cache memory, a determination is made whether the data has
been modified by the processor. If the data has been modified, the
data must be rewritten to lower level memory. To free the second
level cache memory for storage of the newly requested data, the
15 data to be flushed is loaded into a flush buffer for storage during
the rewriting process.